1 Elektronisches Bargeld, ein erstes Beispiel

Auch im Zeitalter des bargeldlosen Bezahlens besitzt das klassische Bargeld durchaus noch seine Berechtigung. Es ermöglicht eine einfache, schnelle, unverbindliche und kostengünstige Abwicklung des Bezahlvorgangs. Bei hohen Beträgen wird Bargeld wegen des Verlust- und Diebstahlrisikos selten verwendet. Hier bietet der bargeldlose Zahlungsverkehr klare Vorteile. Wegen der Abwicklung über eine Bank oder ein Kreditkarteninstitut und der damit verbundenen Dokumentation kann ein derartiger Bezahlvorgang später geprüft und rekonstruiert werden, z. B. anhand eines Kontoauszuges.

Eine neue Problemstellung ergibt sich im Electronic Commerce, das heißt beim Bezahlen von Waren, Dienstleistungen oder Informationen, die im Internet angeboten werden. Die Kosten für viele dieser Dienste bewegen sich im Bereich von wenigen Cent (Micro-Payment). Daher ist eine bargeldlose Transaktion wie zum Beispiel eine Überweisung oder die Belastung einer Kreditkarte unrentabel. Auch möchte der Kunde für die einmalige oder seltene Nutzung eines Dienstes eventuell keine persönlichen Daten oder Kontodaten angeben. Hierzu bietet sich das Bezahlen mit elektronischen Münzen an. Der Bezahlvorgang besteht nur aus dem Übertragen von einigen elektronischen Münzen, das heißt Bitfolgen zwischen Kunde und Händler. Wie beim klassischen Bargeld werden zwischen den beiden Partnern Objekte – nämlich elektronische Münzen – ausgetauscht. Gegebenenfalls wird auch Wechselgeld zurückgegeben, allerdings werden Kunde und Händler damit nicht belastet. Wie beim klassischen Bargeld sollte das Bezahlen anonym erfolgen, gleichzeitig aber sicher gegen Betrug sein.

Das Bezahlen mit elektronischen Münzen effizient und sicher zu gestalten, ist eine Aufgabe der modernen Kryptographie. Anhand einiger einfacher Ideen soll nun exemplarisch gezeigt werden, wie die im Buch beschriebenen kryptographischen Protokolle und Algorithmen hierzu verwendet werden. Die technischen Details folgen dann in Kapitel 9, wenn die Voraussetzungen dafür geschaffen sind. Bevor wir uns jedoch auf den faszinierenden, nicht immer ganz einfachen Weg zum Verständnis dieser Techniken machen, wollen wir am Beispiel des elektronischen Bargeldes ohne Theorie einen ersten Eindruck von den teilweise genialen Protokollen und der Mächtigkeit der modernen Kryptographie vermitteln.

Wir werden schrittweise ein Protokoll mit interessanten Eigenschaften vorstellen. Es wurde von David Chaum, dem Gründer der holländischen Firma Digicash entwickelt [Cha85, Cha92] und patentiert.

Die an dem Verfahren beteiligte Bank nennen wir E-Bank und als Zahlungsmittel werden E-Münzen benutzt. Eine solche E-Münze besteht letztlich aus einer (endlichen) Folge von

Bytes, analog zu einem Geldschein, der ein spezielles Stück bedrucktes Papier darstellt. Wir versuchen's zuerst mal ganz naiv:

Protokoll Nr. 1

Die E-Bank erzeugt auf ihrem PC eine Datei mit dem Inhalt: „E-Münze, Wert: 5 €“, wie in Bild 1.1 dargestellt. Dies führt natürlich sofort zur Inflation, wenn die Kunden den Betrag ihrer E-Münzen beliebig ändern.

Protokoll Nr. 2

Wenn die E-Bank jedoch die E-Münze mit einer Unterschrift versieht, die nur sie und kein anderer erstellen kann, so kann der Kunde, der die Münze auf seinem Rechner speichert, den Betrag nicht mehr abändern. Falls er das versucht, wird die digitale Signatur der Bank ungültig.[1] Er kann jedoch immer noch betrügen, indem er einfach beliebig viele Kopien der E-Münze erzeugt (Bild 1.1). Dies wird verhindert durch Protokoll Nr. 3.

Protokoll Nr. 3

Wie in Bild 1.1 dargestellt, vergibt die Bank nun für jede Münze eine eindeutige Seriennummer und signiert den gesamten Text bestehend aus Betrag und Seriennummer[2]. Versucht nun jemand, Kopien einer derartigen Münze herzustellen, so wird der Betrug erkannt. Die E-Bank protokolliert nämlich in einer zentralen Datenbank alle eingegangenen Seriennummern und sobald mindestens zwei Münzen mit der gleichen Seriennummer zur E-Bank zurückkommen werden Hausdetektiv und Staatsanwalt benachrichtigt.

Dieses Protokoll ist sicher, denn jeder Betrug wird erkannt. Es hat aber noch eine Schwäche. Die Anonymität ist nicht gewährleistet, denn die Bank kann aufgrund der Seriennummern ein perfektes Profil jedes Kunden erstellen (siehe Bild 1.2). Das Problem wird offensichtlich durch die Seriennummern verursacht, auf die wir jedoch aus Sicherheitsgründen nicht verzichten können.

Protokoll Nr. 4

Den Ausweg aus dem Dilemma lieferte David Chaum [Cha85] mit den von ihm erfundenen *blinden Signaturen*. Wie in Bild 1.1 dargestellt, erzeugt nun der Kunde seine E-Münzen selbst. Um eine gültige 5-€-E-Münze zu erhalten, generiert sein PC hundert Dateien, in die jeweils der Text „5 €“ sowie eine große zufällig erzeugte Seriennummer geschrieben werden. Die Seriennummer muss so groß sein, dass die Wahrscheinlichkeit für das zufällige Erzeugen von zwei gleichen Nummern (weltweit) sehr klein ist. Nun bittet er die Bank, eine dieser hundert Münzen blind, das heißt ohne Erkennen von Betrag und Seriennummer, zu signieren. Die Bank wird natürlich nur dann blind signieren, wenn sie sicher ist, dass der Betrag auf der Münze wirklich 5 € ist. Daher wählt sie zufällig 99 der 100 Münzen,

[1] Dies ist ganz analog zu einem unterschriebenen Vertrag, der nicht mehr abgeändert werden *darf*. Bei digitalen Unterschriften ist das Ändern jedoch nicht mehr *möglich*.

[2] In realen Implementierungen wird die Bank weitere Informationen, wie z. B. den Namen der Bank und das Datum, auf der E-Münze speichern. Wir beschränken uns hier jedoch auf die zum Verständnis wesentlichen Daten.

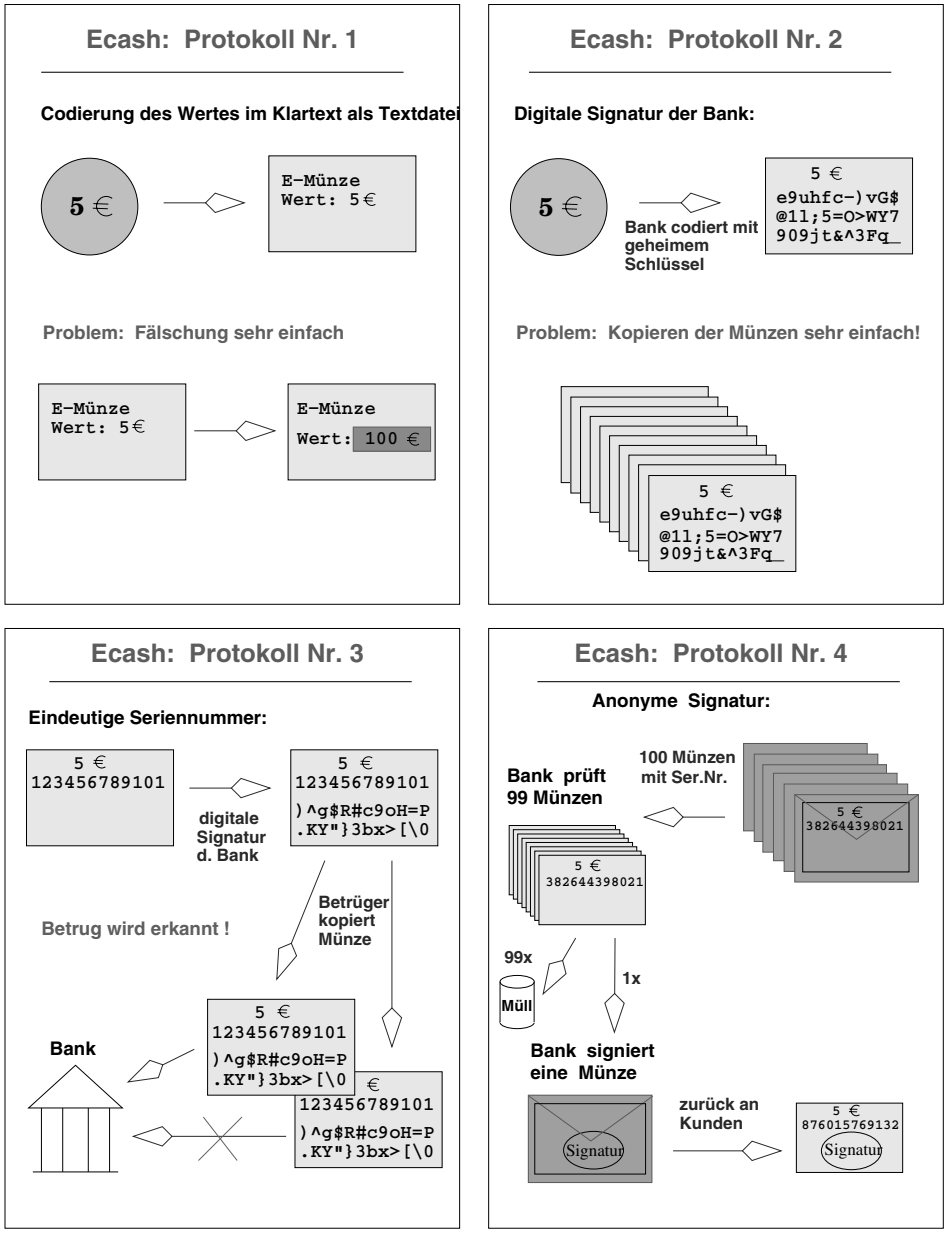

BILD 1.1 Protokolle zum Erzeugen einer E-Münze

die der Kunde auspacken und offenlegen muss. Falls der Betrag 99 mal stimmt, signiert die Bank die letzte Münze blind. Das hierzu benutzte Verfahren verwendet zahlentheoretische Eigenschaften von asymmetrischer Verschlüsselung, die in Kapitel 9 beschrieben werden. Daher beschreiben wir hier das Verfahren nur grob in Analogie zu Geldscheinen aus bedrucktem Papier.

Seriennummer	Ausgabe	Kunde	Kto.-Nr.	Händler	Rücklauf	Betrag
123456789101	12.2.2001	Maier	7654321	Otto Versand	14.2.2001	50 €
123456789102	12.2.2001	Maier	7654321	Otto Versand	14.2.2001	20 €
123456789103	12.2.2001	Maier	7654321	Otto Versand	14.2.2001	8 €
123456789104	12.2.2001	Maier	7654321	Otto Versand	14.2.2001	0.90 €
123456789105	12.2.2001	Maier	7654321	amazon.de	17.2.2001	20 €
123456789106	12.2.2001	Maier	7654321	amazon.de	17.2.2001	2 €
123456789107	15.2.2001	Huber	0054322	Frisör Kurz	15.2.2001	20 €
123456789108	15.2.2001	Huber	0054322	Frisör Kurz	15.2.2001	20 €
123456789109	15.2.2001	Huber	0054322	Frisör Kurz	15.2.2001	5 €
123456789110	15.2.2001	Huber	0054322	Frisör Kurz	15.2.2001	1 €
123456789111	15.2.2001	Huber	0054322	Tankst. Sprit	16.2.2001	100 €
123456789112	15.2.2001	Huber	0054322	Tankst. Sprit	16.2.2001	2 €
123456789113	15.2.2001	Huber	0054322	Tankst. Sprit	16.2.2001	2 €
⋮	⋮	⋮	⋮	⋮	⋮	⋮

BILD 1.2 Beispiel einer möglichen Datenbank von Transaktionen der Kunden der E-Bank

Der Kunde erstellt also 100 Fünfeuroscheine mit Betrag und Seriennummer, packt jeden in einen eigenen Umschlag und legt in den Umschlag über den Geldschein ein Kohlepapier. Die Bank signiert nun den von ihr ausgewählten Geldschein blind, indem sie ihren Stempel aus dem Tresor holt und den Geldschein durch den Umschlag stempelt. Das Kohlepapier hinterlässt auf dem Schein dann den Stempelabdruck. Der Kunde erhält den signierten (gestempelten) Geldschein zurück, packt ihn aus und kann nun damit einkaufen gehen, ohne dass die Bank eine Chance hat, seine Einkäufe zu überwachen. Der Kunde oder auch der Händler kann versuchen, die gültige Münze zu kopieren. Die Bank wird jedoch den Betrug erkennen, weil sie die Seriennummern aller eingehenden Münzen mit den schon eingegangenen in ihrer Datenbank vergleicht. Das Protokoll ist nun also anonym und sicher zugleich.

Ein kleines Problem bleibt jedoch noch zu lösen. Versucht nämlich der Kunde oder der Händler Betrug durch Kopieren der E-Münze, so weiß die Bank zwar, dass der Betrug versucht wurde. Sie weiß jedoch nicht, wer der Betrüger war. David Chaum hat aber auch dieses Problem durch eine elegante Verfeinerung des Protokolls gelöst, die jedoch erst in Kapitel 9 beschrieben werden kann. Hier sei nur so viel verraten: Kopiert der Kunde den Geldschein, so legt die Bank beide eingegangenen Geldscheine übereinander, hält sie gegen das Licht und kann nun den Namen des Betrügers lesen. Ein Geldschein alleine verrät jedoch nichts über die Identität seines Erzeugers.

Übungen

Aufgabe 1.1

a) Ein Betrüger möchte eine Bank, die Protokoll Nr. 4 benutzt, dazu bringen, blind eine 100-€-Münze zu signieren, seinem Konto aber nur einen Euro zu belasten.

Dazu erzeugt er 99 Münzen vom Wert 1 € und eine 100-€-Münze. Wie groß ist die Wahrscheinlichkeit dafür, dass die Bank blind die 100-€-Münze signiert?

b) Wie kann die Bank verhindern, dass der Kunde einen Betrugsversuch unternimmt?

Aufgabe 1.2

Wie viele Bit muss die zufällig generierte Seriennummer einer E-Münze lang sein, damit die Wahrscheinlichkeit für eine zufällige Übereinstimmung von zwei Nummern kleiner ist als die Wahrscheinlichkeit, bei zwei aufeinander folgenden Ziehungen im Lotto (6 aus 49) sechs Richtige zu tippen? Tipp: Berechnen Sie zuerst die Wahrscheinlichkeit, mit einer zufällig erzeugten Seriennummer eine vorgegebene Zahl fester Länge zu treffen. Bestimmen Sie dann deren Länge n. In Abschnitt 6.1.2 wird gezeigt, dass die Seriennummer doppelt so lang (d. h. $2n$) sein muss, um eine gleich geringe Wahrscheinlichkeit für eine zufällige Übereinstimmung von zwei beliebigen Nummern zu erreichen.

2 Grundlagen

2.1 Terminologie

Wie jede Wissenschaft besitzt auch die Kryptographie eine eigene Sprache, deren wichtigste Vokabeln hier kurz vorgestellt werden. Die Begriffe Kryptographie und Kryptologie werden in der Literatur unterschiedlich definiert. Am gebräuchlichsten ist folgende Einteilung: **Kryptographie** wird verstanden als die Lehre der Absicherung von Nachrichten durch Verschlüsseln. **Kryptanalyse** ist die Kunst, Chiffretext aufzubrechen, d. h. den Klartext zu reproduzieren, ohne Kenntnis des Schlüssels. **Kryptologie** vereinigt Kryptographie und Kryptanalyse.

Bei der **Steganographie** werden geheime Nachrichten nicht verschlüsselt, sondern versteckt. Historisches Beispiel hierfür sind unsichtbare Geheimtinten, die später durch Erwärmen sichtbargemacht werden können. Heute werden digitale Daten in den niederwertigen Bits der Farbinformation von digitalen Bildern versteckt. Auch Audiodateien eignen sich aufgrund ihres Rauschens für die Steganographie. Wegen der geringen praktischen Bedeutung wird hier nicht auf die verwendeten Techniken eingegangen.

Ein **Alphabet** A ist eine endliche Menge von Zeichen. $n = |A|$ ist die Mächtigkeit des Alphabets. Der lesbare Text einer Nachricht (message) wird **Klartext** (plaintext) genannt und mit M bezeichnet. Er wird als Zeichenkette über dem Alphabet A gebildet. Zum Beispiel sind aaa und $abcabbb$ Klartexte über $\{a, b, c\}$. **Geheimtexte** oder **Chiffretexte** sind Zeichenketten über dem gleichen Alphabet A oder einem anderen Alphabet. Auch die **Schlüssel** sind Zeichenketten.

Verschlüsselung oder Chiffrierung bezeichnet das Verfahren, um eine Nachricht unverständlich zu machen. Die **Chiffre** E (encryption) ist eine invertierbare, d. h. eine umkehrbare Abbildung, welche aus dem Klartext M und einem Schlüssel K den Geheimtext C (ciphertext) erzeugt. Voraussetzung für die Umkehrbarkeit einer Abbildung ist die Injektivität[1]. Die Umkehrung von E zur Wiederherstellung des Klartextes wird **Entschlüsselung** genannt und mit D (decryption) bezeichnet.

Entsprechend dieser Definitionen gilt $E(M) = C$ und $D(C) = M$, woraus

$$D(E(M)) = M$$

folgt, denn nach dem Entschlüsseln eines Chiffretextes sollte der Klartext zum Vorschein kommen. Praktisch alle kryptographischen Verfahren haben die Aufgabe, eine der folgenden vier Eigenschaften von Nachrichten zu gewährleisten.

[1] Eine Abbildung $f: D \to B$ heißt injektiv, wenn für jedes Paar $x_1, x_2 \in D$ gilt: $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$, d. h. zwei verschiedene Zahlen werden durch f nie auf den gleichen Wert abgebildet.

Geheimhaltung: Ziel der Geheimhaltung ist es, das Lesen einer Nachricht für Unbefugte unmöglich bzw. schwierig zu machen.

Authentifizierung oder Authentifikation: Identitätsbeweis des Senders einer Nachricht gegenüber dem Empfänger, d. h. der Empfänger kann sicher sein, dass die Nachricht nicht von einem anderen (unbefugten) Absender stammt.

Integrität: Die Nachricht darf während der Übermittlung nicht (von Unbefugten) verändert werden. Sie bewahrt ihre Integrität, das heißt ihre Unverletztheit.

Verbindlichkeit: Der Sender kann später nicht leugnen, eine Nachricht abgeschickt zu haben.

2.2 Kryptographische Algorithmen

Kryptographische Algorithmen sind Berechnungsvorschriften, d. h. mathematische Funktionen zur Ver- und Entschlüsselung. Bei **symmetrischen Algorithmen** wird zum Chiffrieren und zum Dechiffrieren immer der gleiche Schlüssel K benutzt und es gilt

$$\begin{aligned} E_K(M) &= C \\ D_K(C) &= M \\ D_K\left(E_K(M)\right) &= M\,. \end{aligned}$$

Bei **asymmetrischen Algorithmen** wird zum Chiffrieren ein Schlüssel K_1 und zum Dechiffrieren ein anderer Schlüssel K_2 benutzt und es gilt:

$$\begin{aligned} E_{K_1}(M) &= C \\ D_{K_2}(C) &= M \\ D_{K_2}\left(E_{K_1}(M)\right) &= M\,. \end{aligned}$$

Man unterscheidet bei kryptographischen Algorithmen zwischen **Stromchiffren** und **Blockchiffren**. Bei Stromchiffren wird ein Zeichen nach dem anderen verschlüsselt. Bei Blockchiffren wird die Nachricht in Blöcke (z. B. der Länge 64 Bit) zerteilt und dann ein Block nach dem anderen verschlüsselt. Die Vereinigung von Algorithmus, zugehörigen Schlüsseln und den verschlüsselten Nachrichten wird **Kryptosystem** genannt.

Früher wurden so genannte **eingeschränkte Algorithmen** benutzt. Bei diesen hängt die Sicherheit davon ab, ob die Arbeitsweise des Algorithmus geheim ist. Die Geheimhaltung eines Algorithmus hat folgende schwerwiegenden Nachteile beim praktischen Einsatz:

- Verlässt eine Person eine Benutzergruppe (z. B. eine Firma), dann muss der Algorithmus geändert werden.
- Auch wenn der Quellcode der Programme nicht öffentlich bekannt ist, kann ein Angreifer aus den Maschinenprogrammen die Algorithmen rekonstruieren. Eingeschränkte Algorithmen können daher nicht an Dritte weitergegeben werden. Sie wären dann wertlos.
- Qualitätskontrolle von eingeschränkten Algorithmen findet in den meisten Fällen nicht in ausreichendem Maße statt, da die entwickelte Software nicht der Kritik und den Angriffen der Öffentlichkeit standhalten muss.

Heute werden Algorithmen mit **Schlüssel** benutzt. Der Schlüssel ist meist eine natürliche Zahl, dargestellt im Binärsystem, d. h. als Folge von Bits. Der Algorithmus ist idealerweise allgemein bekannt und nur der zugehörige Schlüssel muss geheim gehalten werden. Dieses Vorgehen wurde schon im 19. Jahrhundert von A. Kerkhoffs [Kah67] gefordert:

> Die Sicherheit eines Verschlüsselungsverfahrens darf nur von der Geheimhaltung des Schlüssels abhängen, nicht jedoch von der Geheimhaltung des Algorithmus.

Kerkhoffs forderte damit, dass die Sicherheit eines Algorithmus nicht darunter leiden darf, dass er veröffentlicht wird. Die aktuelle Praxis in der Kryptographie zeigt deutlich, dass durch möglichst frühzeitige Offenlegung der Algorithmen die Sicherheit eines Kryptosystems erheblich größer wird. Denn sobald ein Algorithmus publiziert ist, muss er den Attacken der Experten standhalten, d. h. er muss sich bewähren. Sind über einen langen Zeitraum alle Attacken erfolglos, so stärkt dies das Vertrauen der Benutzer in die Sicherheit des Algorithmus. Diese Methodik der Entwicklung moderner Algorithmen ist ein wichtiger Bestandteil der so genannten **starken Kryptographie**.

In der Geschichte der Kryptographie gibt es viele Beispiele für die Verletzung von Kerkhoffs' Prinzip, was zu teilweise dramatischen Sicherheitslücken führte. Zwei Beispiele aus dem Jahr 1999 zeigen, dass selbst namhafte Firmen das Kerkhoffs-Prinzip nicht beachten. Im Online-Magazin der Zeitschrift c't vom 7.12.99[2] war folgender Text zu lesen:

> **Handy-Verschlüsselung angeblich geknackt**
>
> Die beiden israelischen Kryptologen Alex Biryukov und Adi Shamir haben Medienberichten zufolge den Verschlüsselungsalgorithmus geknackt, der GSM-Handy-Telefonate auf der Funkstrecke zur Mobiltelefon-Basisstation schützt. ...
>
> Eines zeigen die Vorfälle um die GSM-Verschlüsselungsalgorithmen A5/1 und A5/2 aber schon jetzt deutlich: *Der Versuch, Krypto-Verfahren geheim zu halten, dient nicht der Sicherheit.* Das hat anscheinend auch die GSM-Association gelernt: Ihr Sicherheitsdirektor James Moran äusserte dem Online-Magazin Wired gegenüber, dass man künftige Algorithmen von vorneherein offenlegen will, um der Fachwelt eine Prüfung zu ermöglichen. (nl/c't)

Eine Woche später, nämlich am 15.12.99[3] erschien an gleicher Stelle die nächste Meldung zu diesem Thema:

> **Netscape verschlüsselt Passwörter unzureichend**
>
> Der Netscape Navigator legt Passwörter für den Zugriff auf E-Mail-Server nur unzureichend verschlüsselt ab. Zwei Mitarbeiter des US-Softwarehauses Reliable Software Technologies (RST) brauchten lediglich acht Stunden, um den Algorithmus zu knacken. ...
>
> Der Algorithmus zerhacke die Passwörter zwar, es handle sich jedoch um *keine starke Verschlüsselung*, so Gary McGraw von RST. Durch die Eingabe einfacher Passwörter wie „a", „b" und so weiter sei man relativ schnell dahinter gekommen.
>
> ...
>
> Der US-Sicherheitsexperte Bruce Schneier wertet die Entdeckung als weiteres Beispiel dafür, *wie schädlich proprietäre Verschlüsselungsverfahren sein können.* (ad[2]/c't)

[2] Siehe http://www.heise.de/newsticker/data/nl-07.12.99-000/
[3] Siehe http://www.heise.de/newsticker/data/ad-15.12.99-001/

Ein weiteres aktuelles Beispiel betrifft das Verschlüsselungsprotokoll WEP (Wired Equivalent Privacy), das bei Funk-Netzwerken nach dem Standard IEEE802.11 verwendet wird. Die Autoren von [BGW01] schreiben

> **Conclusions**
>
> Wired Equivalent Privacy (WEP) isn't. The protocol's problems is a result of misunderstanding of some cryptographic primitives and therefore combining them in insecure ways. These attacks point to *the importance of inviting public review* from people with expertise in cryptographic protocol design; had this been done, the problems stated here would have surely been avoided.

Diese drei Meldungen sprechen für sich und bedürfen keines weiteren Kommentars.

2.3 Kryptographische Protokolle

Ein kryptographischer Algorithmus zum Verschlüsseln kann auf vielfältige Art und Weise in unterschiedlichen Anwendungen eingesetzt werden. Damit eine Anwendung immer in der gleichen und korrekten Art abläuft, werden kryptographische Protokolle definiert.

Im Gegensatz zu den kryptographischen Algorithmen handelt es sich bei den Protokollen um Verfahren zur Steuerung des Ablaufs von Transaktionen für bestimmte Anwendungen, wie zum Beispiel das in Kapitel 1 vorgestellte Protokoll für elektronisches Bargeld.

2.4 Public-Key-Algorithmen

Wollen zwei Parteien über einen unsicheren Kanal mit einem symmetrischen Algorithmus geheime Nachrichten austauschen, so müssen sie einen geheimen Schlüssel vereinbaren. Wenn sie nur über einen unsicheren Kanal verfügen, sind sie mit dem Schlüsseltauschproblem (Kapitel 5) konfrontiert.

Erst Mitte der 70er Jahre wurde mit der Erfindung der Public-Key-Kryptographie eine befriedigende Lösung gefunden. Sie kam genau zum richtigen Zeitpunkt, um für eine sichere Kommunikation im Internet den Grundstein zu legen. Systeme wie zum Beispiel PGP [Zim95a] (Kapitel 8.1) zum Verschlüsseln von E-Mails wären undenkbar ohne Public-Key-Algorithmen.

Vor der Erfindung der Public-Key-Algorithmen beschränkte sich das Verschlüsseln von Nachrichten auf spezielle, zum Beispiel militärische Anwendungen, bei denen der hohe Aufwand für den Schlüsseltausch gerechtfertigt war. Mit Hilfe der Public-Key-Kryptographie kann nun jedermann mit beliebigen Partnern geheime Nachrichten austauschen, Dokumente signieren und viele andere kryptographische Anwendungen wie zum Beispiel elektronisches Bargeld nutzen.

Algorithmen mit öffentlichem Schlüssel sind asymmetrische Algorithmen, die einen geheimen Schlüssel S (secret key) sowie einen öffentlichen Schlüssel P (public key) benutzen, deren Arbeitsweise und Sicherheit in Kapitel 5 ausführlich untersucht wird.

Die Idee der Public-Key-Kryptographie ist in Bild 2.1 dargestellt. Wenn Bob[4] geheime Botschaften empfangen möchte, so erzeugt er einen öffentlichen Schlüssel P_B, den er all seinen Kommunikationspartnern zukommen lässt und einen geheimen Schlüssel S_B, den er sicher verwahrt.[5]

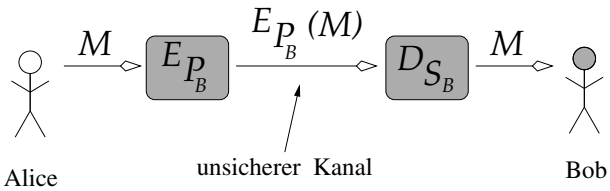

BILD 2.1 Austausch einer Nachricht mit einem Public-Key-Verfahren. Es werden öffentlicher Schlüssel P_B und geheimer Schlüssel S_B von Bob benutzt

Will nun Alice eine geheime Nachricht an Bob schicken, so benutzt sie zum Verschlüsseln den öffentlichen Schlüssel P_B von Bob. Dieser dechiffriert die Nachricht dann mit seinem geheimen Schlüssel S_B. Zum Verschlüsseln wird nur der öffentliche Schlüssel benötigt. Mit ihm kann also jedermann eine verschlüsselte Nachricht an Bob schicken, aber nur Bob kann sie mit seinem geheimen Schlüssel lesen. Dieses Prinzip entspricht der Funktion vieler Wohnungstüren, bei denen das Schloss verriegelt, sobald die Türe geschlossen wird. Jedermann kann die Türe schließen. Das Öffnen von außen ist dagegen nur für den Besitzer des Schlüssels möglich.

Damit Bob auch tatsächlich den Original-Klartext liest, muss gelten:

$$\begin{aligned} E_{P_B}(M) &= C \\ D_{S_B}(C) &= M \\ D_{S_B}\left(E_{P_B}(M)\right) &= M\,. \end{aligned}$$

Beim Signieren eines Dokumentes M geht man umgekehrt vor wie beim Verschlüsseln. Im Prinzip verschlüsselt Alice das Dokument mit ihrem geheimen Schlüssel und hängt das Resultat als Signatur an das Dokument an. Wenn nun am Dokument oder an der Signatur auch nur ein Bit geändert wird, ist die Signatur ungültig (Kapitel 6).

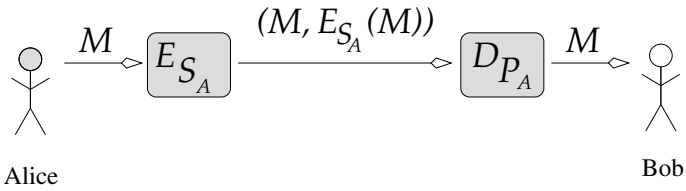

BILD 2.2 Alice signiert ein Dokument M mit ihrem geheimen Schlüssel S_A und Bob prüft die Signatur mit Alices öffentlichem Schlüssel P_A

[4] „Alice“ und „Bob“ als Kommunikationspartner sind Bestandteil der kryptographischen Fachsprache.

[5] Zur Vermeidung von Missverständnissen sei hier schon bemerkt, dass der Empfänger eines öffentlichen Schlüssels P_B immer dessen Authentizität überprüfen muss (Kapitel 7).